# COMPUTER IMPLEMENTED SECRET OBJECT KEY BLOCK CIPHER ENCRYPTION AND DIGITAL SIGNATURE DEVICE AND METHOD



Inventor: Luciano F. Paone
18 Knobhill Drive
Kings Park, New York 11754

# COMPUTER IMPLEMENTED SECRET OBJECT KEY BLOCK CIPHER ENCRYPTION AND DIGITAL SIGNATURE DEVICE AND METHOD

## COPYRIGHT NOTICE





## BACKGROUND OF THE INVENTION

### Field Of The Invention

This invention relates generally to a computer implemented device and method for cryptography and, more particularly relates to a computer implemented block cipher encryption method utilizing dynamic device keys and a digital signature.

## DESCRIPTION OF THE PRIOR ART

The principal goal of encryption is to render communicated data secure from unauthorized eavesdropping. This is generally referred to as the "secrecy" or "confidentiality" requirement of cryptographic systems. A related requirement is the "authenticity" or "integrity" requirement, which ensures that the communicated information is authentic, i.e. that it has not been tampered with, either deliberately or inadvertently. For purposes of further discussion, some definitions are provided.

"Plaintext" is used to refer to a message before encrypting and after decrypting by a cryptographic system. "Ciphertext" is the form that the encrypted part of the message takes during transmission over a communications channel or within a computer memory storage device. "Encryption" is the process of transformation from plaintext to ciphertext. "Decryption" is the process of transformation from ciphertext to plaintext. Both encryption and decryption are controlled by keys. Without knowledge of the encryption key, a message cannot be encrypted, even with knowledge of the encrypting process. Similarly, without knowledge of the decryption key, the message cannot be decrypted, even with knowledge of the decrypting process.

Data encryption processes scramble plaintext data into ciphertext to prevent unauthorized access to the data. Decryption processes restore the plaintext from the ciphertext (encrypted data). Symmetric key encryption processes utilize the same key for encryption and decryption.

In order to ensure the integrity of the ciphertext, the encryption must be sufficiently complex to prevent unauthorized access to the encrypted data. Two current methods of cryptanalytic attacks, methods designed to break encryption processes, are linear and differential cryptanalysis. In linear cryptanalysis, linear approximations of the block cipher and key schedule used in the encryption process are constructed. Collected plaintexts and associated ciphertexts are used to exploit a bias in the encryption process and key schedule. If the bias is very small many plaintexts and associated ciphertext pairs are used. This method tries different keys and eventually converges on the correct key. In differential

cryptanalysis, pairs of ciphertexts whose plaintexts have particular differences are compared. The evolution of these differences as the plaintexts propagate through the rounds of the encryption process when they are encrypted with the same key are analyzed. Different probabilities are assigned to different keys. As more and more ciphertext pairs are analyzed, the method converges on the correct key. Both cryptanalytic attacks exploit the fact that existing symmetric key block cipher encryption processes use static keys to create the ciphertext.

Still other cryptanalytic attacks exploit the fact that current encryption processes use small key spaces and small block sizes. For certain key spaces and block sizes, current data storage technologies now make it possible to store all combinations of an encrypted block with an associated key for a chosen plaintext block. Thus, breaking the encryption process merely involves a quick look-up table. Accordingly, the present invention seeks to overcome the disadvantages associated with currently available encryption methods and create ciphertext which is very secure and substantially immune to known cryptanalytic attacks.

## OBJECTS AND SUMMARY OF THE INVENTION

It is an object of the present invention to provide a computer implemented encryption device and method which is substantially immune to currently available cryptanalytic attacks using an object key comprised of data and methods which operate on the data.

It is another object of the present invention to provide a computer implemented block cipher encryption device and method using an object key which is dynamic, i.e., changing throughout the encryption process.

It is still a further object of the present invention to provide a computer implemented block cipher encryption device and method using a dynamic object key which is modified by a random session object key.

It is yet another object of the present invention to provide a computer implemented block cipher encryption device and method such that each input plaintext data block is encrypted using a new key schedule to create the ciphertext.

It is a further object of the present invention to provide a computer implemented encryption device and method which includes a digital signature appended to the ciphertext, the digital signature being unique to the data being signed.

It is yet another object of the present invention to provide a computer implemented encryption device and method using a secret key composed of two 2048-bit user object keys and 512-bit random session object key.

The present invention provides a computer implemented data encryption device and method utilizing object keys (consisting of data and methods that operate on the data), a large key space and a large block size. The object keys (K_OBJECT) are dynamic keys and are

composed of a 4096-bit static initial state that is created by the user and a method that modifies the keys based on seeding from a random session object key (R_OBJECT). The key modification is performed for each input plaintext data block so that each data block is encrypted with a different key. The initial state of the object key is used in the block cipher encryption process to encrypt a 512-bit random session key. The random session object key is used as the initial state of the random session object key (R_OBJECT). The running states of the random session object key (R_OBJECT) seed the object key (K_OBJECT) modification methods. The object key's (K_OBJECT) running state provides an index for seeding of the random session object key's (R_OBJECT) modification method. The encryption process utilizes a 512-bit (64 byte) input block size.

The object key (K_OBJECT) is preferably composed of two 2048-bit sub-object keys (K1 and K2). The two sub-object keys are used as inputs to an expansion function that provides 13584 bytes for the block's key schedule. Similar to the object key (K_OBJECT), the block's key schedule is regenerated anew for each plaintext input block.

The encryption process includes many linear and nonlinear-keyed operations. The keyed operations include addition, 32-bit sliding window rotation, bit-wise exclusive or, 8-bit by 8-bit substitution using different transverse counts for each substitution, byte transposition and bit transposition. All operations are nested and repeated multiple times.

The decryption process is identical to the encryption process with the exception that the keyed encryption operations on the data are run in reverse.



6

In order to authenticate an encrypted file, a digital signature is provided. To create the digital signature, the ciphertext is used as input into a 2048-bit object keyed one-way hash function to produce a 2048-bit digital signature that is appended to the ciphertext. The 2048-bit digital signature object key is seeded with the output of hash rounds. A third party or secured server can execute verification software that contains the user's secret digital signature key to regenerate the digital signature of the ciphertext that was signed and compare it to the signature that is appended to the ciphertext to determine if they match. The plaintext is not compromised and only the party that contains the receiver's encryption key can create ciphertext that the receiver can decrypt into plaintext.

A preferred form of the encryption device and method, as well as other embodiments, objects, features and advantages of this invention will be apparent from the following detailed description of illustrative embodiments thereof, which is to be read in connection with the accompanying drawings.

## BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a simplified block diagram of a computer based system capable of implementing the encryption method of the present invention.

Figure 2 is a simplified flow chart representation of the installation and initialization of the encryption software onto a computer based system of Figure 1.



7

Figure 3 is a simplified flow chart representation of the encryption process performed by the computer system formed in accordance with the present invention.

Figure 4 is a simplified flow chart representation of creating the digital signature to be appended to ciphertext created by the encryption process of Figure 3.

Figure 5 is a flow chart representation of the steps carried out by the computer system to create ciphertext.

Figure 6 is a flow chart representation of the switch key function formed in accordance with the present invention.

Figure 7 is a flow chart representation of the modification function of the sub-object key K1 based on seeding from the random session object key K3.

Figure 8 is a flow chart representation of the modification function of the sub-object key K2 based on seeding from the random session object key K3.

Figure 9 is a flow chart representation of the methods to create unique key schedules for each block of plaintext to be encrypted in accordance with the present invention.

Figure 10 is a flow chart representation of the method to create a digital signature for each encrypted file formed in accordance with the present invention.

Figure 11 is a flow chart representation of the modification method of the object key used in generating a digital signature formed in accordance with the present invention.

Figure 12 is a flow chart representation of the method of producing a digital signature in accordance with the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring initially to Figure 1, a simplified block diagram of a computer based system 10 used for implementing the encryption method of the present invention is illustrated. In general, the encryption method of the present invention may be implemented in software and embodied on a magnetic storage device 2 such as a computer diskette. The diskette may be inserted and into and read by a disk drive 4 for execution by the computer system 10. The computer system 10 generally includes a display device 6, such as a CRT display, a keyboard 8 for entering information, a pointing device 12, and a printer 14. Internally, the computer system 10 includes a central processing unit (CPU) 16 which preferably includes an internal clock function, a memory device, such as a random access memory (RAM) and an output port which may be connected to a modem for remote access to other computer systems within a network. The various computer system components are operatively coupled and communicate via a system bus 24, or similar hardware architecture.

It will be appreciated by those of ordinary skill in the art that the computer system illustrated in Figure 1 is merely representative of one form of computer based system capable of carrying out the functions of the encryption method of the present invention. For example,

the encryption method may be executed on a single computer workstation to protect information stored therein or in a cryptographic communications system including a plurality of computer stations cooperatively coupled and operating via a computer network to protect information being transmitted and received via the network.

Further, it is to be understood that the individual system components may vary in type, while still providing similar functions. For example, the pointing device 12, used to position a display cursor and select certain functions and/or items displayed on the display device may be in the form of a mouse, trackball or touchscreen. In a preferred embodiment, the user is provided with a keyboard 8 for data entry and key initialization and a mouse as the pointing device 12.

Regardless of the computer system being used, the encryption method of the present invention may be loaded from the diskette into the disk drive 4 and executed by the CPU 16 in conjunction with the RAM 18. In this manner, the functions of the encryption method can be executed to create ciphertext (encrypted data) from input plaintext, or to decrypt ciphertext to restore the input plaintext.

Referring now to Figure 2, the installation and initialization of the encryption method is illustrated in flow chart format. The encryption software is initially installed 26 into the disk drive 4 and executed by the CPU 16 and RAM 18. Once installed, the user types a series of random key strokes 28. Based upon the random keystrokes, an initial state of the object key (K_OBJECT comprising sub-object key K1 and sub-object key K2) is created 30. At this



10

point, the user creates a password 32 which becomes associated or linked with the initial state of the object key. The initial state of the object key is appended with a checksum and encrypted 34 along with the user's password. The user then creates a password for a remote user 36 to allow decryption of transmitted ciphertext. The remote user password is also appended with a checksum and encrypted 38. With this initialization procedure completed, the user is ready to create ciphertext from plaintext data using the encryption method of the present invention.

The computer implemented encryption device and method of the present invention utilizes object keys, consisting of data and methods that operate on the data, a large key space and a large block size. More specifically, the present invention discloses a 4096-bit secret object key block cipher encryption method utilizing a 2048-bit object keyed digital signature process to sign encrypted files. In the object key, the data comprises a set of binary digits that can represent two to the power of the bit length possible combinations. The encryption method uses a 512-bit block and the object key is a dynamic key. The object key (K_OBJECT) contains key modification methods that mutate the key's state based on seeding from a random session object key (R_OBJECT). The random session object key also consists of data and methods that operate on the data. As illustrated in Figure 1, the user of the encryption method creates the object key's 4096-bit initial state.

In order to create an encrypted file, the input plaintext file is compressed using a redundant byte reducing method and padded with random bytes to produce a file with a length being a multiple of 512-bits (64 bytes) 40. Using a time clock which is a part of the

computer system CPU, a 512-bit random number is generated and assigned as an initial state of the random session object key (R_OBJECT) 42.

Next, a switch key is created 44, from an initial state of the object key (K_OBJECT comprising K1 and K2). The function of the switch key will be discussed later in further detail. In the block cipher encryption method in accordance with the present invention, the plaintext blocks are encrypted using a dynamic key schedule. The encryption key schedule is created 46 from an initial state of the object key (K_OBJECT comprising K1 and K2). As will become apparent, the encryption method of the present invention encrypts each plaintext data block utilizing a different key schedule to produce ciphertext that is immune to current cryptanalytic attacks designed to work with static key schedules.

Using the initial state of the key schedule, the initial state of the random session object key is encrypted 48. At this point in the encryption method of the present invention, the random session object key is modified 50 based on seeding from the object key (K_OBJECT using K2 only). This modification of the random session object key is the first step within a logic loop to determine when the entire plaintext file has been transformed into ciphertext. In the next step, the state of the object key (K_OBJECT comprising K1 and K2) is modified based on seeding from the random session object key 52. Using the modified object key from the previous step, a new key schedule is created 54. The new key schedule is utilized to encrypt the input plaintext data block using the modified object key (K_OBJECT comprising K1 and K2) 56. At this point, the encryption method includes a decision box to determine if further plaintext data blocks exist to be encrypted 58. If further plaintext data blocks exist,

the method returns to the step in which the random session object key is modified based on seeding from (K_OBJECT using K2 only) 50. Accordingly, new states for the random session object key (step 50), the object key (K_OBJECT comprising K1 and K2) (step 52), and a new key schedule (step 54) are created for each plaintext data block which is encrypted. Once all plaintext data blocks are encrypted, the encryption method of the present invention transposes 60 the encrypted data using the switch key created in step 44. More specifically, a final 128 byte transposition is performed on the encrypted data with the first 128 bytes of ciphertext transpositioned within the entire ciphertext file with respect to the switch key. After transposition, the encryption is completed 62.

In the preferred embodiment of the present invention, a digital signature is produced and appended to the ciphertext. A digital signature allows a third party or secured server to execute verification software that contains the original user's secret digital signature key to regenerate the digital signature of the ciphertext and compare it to the signature that was appended to the encrypted ciphertext to determine if they match. In this manner, the plaintext is not compromised and only the party that has access to the receiver's encryption key can create ciphertext that the receiver can decrypt into plaintext.

The unique method for creating the digital signature is illustrated in the flow chart of Figure 4. The encrypted data file or ciphertext is input into a 2048-bit object keyed one-way hash function 64. A 2048-bit digital signature is produced from the ciphertext along with a 2048-bit object key specific to the particular input file 60. The 2048-bit digital signature is

appended to the encrypted data or ciphertext. The 2048-bit digital signature object key is seeded with the output of hash rounds to create the digital signatures for each plaintext file.

As previously noted, the computer implemented encryption device and method of the present invention preferably uses an object key (K_OBJECT) which comprises two 2048-bit sub-object keys (K1; K2). The two sub-object keys are used as inputs to an expansion function that provides 13,584 bytes for the cipher block's key schedule (KS).

More specifically, Figure 5 is a flow chart representation of the process to create ciphertext using an object key comprising two sub-object keys and a random session object key. In Figure 5, the object key is made up of two 2048-bit sub-object keys K1 and K2, respectively. The random session object key is represented by K3 (K3=srand((KS[i]xKS[i]+KS[1])^ time (NULL); K3[i] =((rand ()%255)+1+High_resolution_timer) where "i" is the running index) and KS is the key schedule used to encrypt each plaintext input block of 64 bytes. As earlier described, the object keys K1 and K2 are composed of initial states that are created by the user and functions that modify the keys for each plaintext input block.

Referring to Figure 5, the input file is compressed using a redundant byte reducing method and padded 72 with random bytes to produce a file with a length having a multiple of 512-bits (64 bytes). The encryption process extensively uses linear and non-linear keyed operations on the plaintext to produce the ciphertext. A substitution array 74 consisting of 256 unique 8-bit elements and a transverse array 76 consisting of 64 unique 8-bit elements

are continuously transpositioned with respect to the current key schedule. The two arrays 74, 76 provide an 8-bit by 8-bit S-box 78 (substitution process) that contains a transverse count substitution process for each input into the S-box. More specifically, the steps 74, 76 and 78 provide a transposition of a sequence of integers 0-63 with respect to a key and provides a count of substitution rounds for a particular input entering the S-box. Continuing the method, a nested keyed addition 80, sliding window rotation 82, bit-wise exclusive or 84, 8-bit by 8-bit transverse repeated substitution 86, 88, byte transposition 90 and bit transposition encryption process 92 are executed by the computer system. The sliding window rotation operates on a 32-bit boundary incrementing (sliding) one byte (8-bits) after each rotation are used to create the ciphertext. The outer loop cycles four times with the inner loop cycling four times for each outer loop iteration. After each block has been processed, the key modification methods in the object key (K_OBJECT) and the random session object key (R_OBJECT) are performed to modify the object key's and the random session object key's state. A new key schedule is created after the modification methods are performed. The key schedule creation process is illustrated in Figure 9. In Figure 9, the two 2048-bit object key states are used to create a 13,584 byte key schedule for the encryption process.

Referring to Figure 9, the key schedule is created with multiple linear congruential generators. Three offset variables are utilized. Offset_one is initialized to 0, offset_two is initialized to 1 and offset_three is initialized to 2. The first byte of key schedule is set to K1[K2[#]] + K2[K1[#]] (Step 136). In Figure 9, the letter "a" is a multiplier, "b" is an offset and "a_prev" is the previous value of "a". In step 138, a_prev is initialized. The first linear congruential generator uses K1[index] as a multiplier and K2[index + offset_one] as an offset

(Step 140). The index is a running counter. The second linear congruential generator uses K2[index + offset_two] as a multiplier and K1[index + offset_three as an offset (step 142). The third linear congruential generator produces the key schedule. The third linear congruential generator uses the output of the first linear congruential generator as a multiplier and the output of the second linear congruential generator as an offset (step 144). Offset_one, offset_two and offset_three are incremented every 256 rounds (steps 146, 148). In the preferred embodiment, the linear congruential generators are run through 13,584 rounds to produce a 13,584 byte key schedule. (KS as shown in Figure 5).

After all the plaintext blocks have been processed a final 128-byte transposition 98 is performed with the first 128 bytes of ciphertext transpositioned within the entire ciphertext file with respect to a switch-key (SWK). As illustrated in Figure 6, the switch-key is created from the initial state of the object key. The switch key is initialized with elements of the initial state of the object key. The switch key is grouped by 32-bit blocks and the current switch key element is replaced using the following steps:

the current switch key element is bit-wise exclusive ored to a switch key element indexed two elements from the current element (step 101);

the output of the previous operation is rotated to the right switch key indexed three elements from the current element modulus thirty-one plus one (step 103);

the output of the previous operation is bit-wise exclusive ored to a switch key element indexed three elements from the current element (step 105); and

the previous three steps are repeated for each final transposition switch operation. In an alternative embodiment, the steps 101, 103, 105 may include a hashing function in the creation of the switch key.

Referring back to Figure 5, the plaintext file extension and a checksum are appended to the ciphertext. Delimiters are used to mark the beginning and end of the ciphertext. This completes the encryption process to create the ciphertext, which, in a preferred embodiment, will include a digital signature appended thereto.

As earlier noted, the object key is dynamic and changes with each block of input data. The first sub-object key's modification method, function F1 of Figure 5, is illustrated in Figure 7. In the first execution of the modification method, the random session key's elements are used instead of the running index "s". The first sub-object key (K1) has a modification method which includes the following iterations:

performing a bit-wise exclusive or on an unsigned byte of the random session object key(K3) to an unsigned byte of the current state of the object key provided by an incremented index into the current state of the object key (I_BYTE_OBJECT_KEY) (step 104);

performing an unsigned byte addition on the output byte of the previous operation (PREV_OUTPUT) with I_BYTE_OBJECT_KEY (step 106);

performing a 16-bit multiplication of PREV_OUTPUT and I_BYTE_OBJECT_KEY modulus 254 and add 2 (step 108);

performing a 16-bit addition of PREV_OUTPUT and I_BYTE_OBJECT_KEY (step 110);

performing another 16-bit addition of PREV_OUTPUT and I_BYTE_OBJECT_KEY (step 112);

performing a bit-wise exclusive or of PREV_OUTPUT with a 16-bit unsigned integer of the current state of the object key provided by an incremented index into the current state of the object key (I_INT_OBJECT_KEY) (step 114);

rotating PREV_OUTPUT to the right I_BYTE_OBJECT_KEY modulus 15 plus 1 times (step 116);

performing a 16-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY (step 118);

performing a 16-bit multiplication of PREV_OUTPUT and I_INT_OBJECT_KEY with the lower order byte of I_INT_OBJECT_KEY modulus 254 plus 2 (step 120);

performing a 16-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY (step 122);

performing another 16-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY (step 124);

performing a bit-wise exclusive or of PREV_OUTPUT with a 32-bit unsigned long integer of the current state of the object key provided by an incremented index into the current state of the object key (I_LONG_INT_OBJECT_KEY) (step 126);

rotating PREV_OUTPUT to the left I_BYTE_OBJECT_KEY modulus 31 plus 1 times (step 128);

performing a bit-wise exclusive or of PREV_OUTPUT with I_LONG_INT_OBJECT_KEY (step 130);

repeating the previous set of operations eighty-four times substituting the random seed unsigned byte with a byte from the four byte output block provided by the previous set of operations recursively setting the current output block to the next output block when the current output block is exhausted and utilizing a different ordered byte each round (step 132);

performing a byte transposition of the bytes in the new 256 byte output block (N_OUTPUT) provided by the previous set of operations utilizing the following operation:

performing a byte-wise index through N_OUTPUT and switching the current byte of N_OUTPUT with the N_OUTPUT byte indexed at position I_BYTE_OBJECT_KEY, indexing through the entire block of N_OUTPUT (step 134). This modification method makes the ciphertext generated using the dynamic object key immune from cryptanalytic attacks.

The second sub-object key's (K2) modification method is illustrated in Figure 8. The second sub-object key's modification method is similar to the first sub-object key's modification method with the exception that the rotations are performed in the reverse. Accordingly, further discussions of the flow chart illustrated in Figure 8 is not necessary.

Referring now to Figure 10, the eight hash functions used to create the 2048-bit object keyed digital signature is illustrated 150. In the preferred embodiment, a digital signature is created and appended to each input plaintext file transformed into ciphertext for authentication purposes. The hash functions illustrated in Figure 10 are one-way hash

functions. Referring to Figure 12, the process for generating the digital signatures for a file under control of an object key using keyed hashing of the input ciphertext blocks includes the following iterations:

dividing the input data into 256 byte data blocks (step 200);

further dividing the data block into 64 32-bit blocks (VAR_BLOCK) (step 202);

modifying each VAR_BLOCK and element of the key by a plurality of unique one-way irreversible hash function (step 202); (It should be noted that before each VAR_BLOCK modification, the object key used to create the digital signature is modified as illustrated in Figure 11 (Steps 205).

repeating the previous steps for all VAR_BLOCK hash function combinations to create an output running message digest (steps 204);

performing a bit-wise exclusive or of the running message digest to the next input data block (step 206);

repeating the previous four steps for all data blocks (step 208);

transpositioning each byte of an output from the previous step by switching a position of each byte with another byte at a position provided by an element of the key wherein the position provided by an element of the key is bounded by the size of the data block (step 210);

appending a checksum to the digital signature (step 212); and

appending the digital signature to the ciphertext (step 214).

Delimiters are used to mark the beginning and end of the digital signature. The digital signature generated is appended to the ciphertext to provide for verification by a third party or secured server as earlier described.

Figure 11 illustrates the digital signature's object key modification method. The digital signature's object key modification method is composed of the following:

performing an 8-bit addition or of a seed to an unsigned byte of a current state of the object key provided by an incremented index into the current state of the object key (I_BYTE_OBJECT_KEY) (step 159);

performing an unsigned byte addition on the output byte of the previous operation (PREV_OUTPUT) with I_BYTE_OBJECT_KEY (step 160);

performing a bit-wise exclusive or of PREV_OUTPUT to I_BYTE_OBJECT_KEY (step 162);

performing a 16-bit multiplication of PREV_OUTPUT and I_BYTE_OBJECT_KEY modulus 254 and add 2 (step 164);

performing a 16-bit addition of PREV_OUTPUT and I_BYTE_OBJECT_KEY (step 166);

performing another 16-bit addition of PREV_OUTPUT and I_BYTE_OBJECT_KEY (step 168);

performing another 16-bit addition of PREV_OUTPUT and I_BYTE_OBJECT_KEY (step 170);

rotating PREV_OUTPUT to the left I_BYTE_OBJECT_KEY modulus 15 plus 1 times (step 172);

performing a 16-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY (step 174);

performing a 16-bit multiplication of PREV_OUTPUT and I_INT_OBJECT_KEY with the lower order byte of I_INT_OBJECT_KEY modulus 254 plus 2 (step 176);

performing a 16-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY (step 178);

performing another 16-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY (step 180);

performing a bit-wise exclusive or of PREV_OUTPUT with a 32-bit unsigned long integer of the current state of the object key provided by an incremented index into the current state of the object key (I_LONG_INT_OBJECT_KEY) (step 182);

rotating PREV_OUTPUT to the right I_BYTE_OBJECT_KEY modulus 31 plus 1 times (step 184);

performing a 32-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY (step 186);

repeating the previous set of operations eighty-four times substituting the seed unsigned byte with a byte from the four byte output block provided by the previous set of operations recursively setting the current output block to the next output block when the current output block is exhausted and utilizing a different ordered byte each round (step 188);

performing a byte transposition of the bytes in the new 256 byte output block (N_OUTPUT) provided by the previous set of operations utilizing the following operation:

performing a byte-wise index through N_OUTPUT and switching the current byte of N_OUTPUT with the N_OUTPUT byte indexed at position I_BYTE_OBJECT_KEY and

finally indexing through the entire block of N_OUTPUT (step 190). This modification method is used to modify the object key for each block of input data. Accordingly, the object key for creating the digital signature is also is dynamic as earlier described.

In the preferred embodiment, the encrypted ciphertext is signed by the originator to authenticate the information. More specifically, a digital signature is generated which is appended to the ciphertext. In the present invention, the digital signature is created under the control of the key, preferably an object key comprising data and methods that modify the data, and is unique to each ciphertext file by using the ciphertext as input into the digital signature generation process.

Accordingly, the present invention overcomes the disadvantages of known encryption processes by creating ciphertext immune to currently available cryptanalytic attacks. The cryptographic communications system of the present invention provides for an encryption process utilizing a dynamic object key. The initial state of the object key is created by the user and a method that modifies the keys based on seeding from the random session object key which is also a dynamic key whose initial state is set by a random number. Based on the object key, a different key schedule is used to encrypt each block of data in the preferred block cipher data encryption process. It will be appreciated by those skilled in the art that the use of a dynamic object key and encrypting with different key schedules can make most currently available encrypting processes stronger, i.e., more immune to cryptanalytic attacks. Tho authenticate the ciphertext, a 2048-bit secret key digital signature is appended to the ciphertext. The digital signature is generated using the ciphertext as input and is unique for

each ciphertext file. More specifically, the ciphertext is sent into a one-way keyed hash function to produce the digital signature.

Although the illustrative embodiments of the present invention have been described herein with reference to the accompanying drawings, it is to be understood that the invention is not limited to those precise embodiments, and that various other changes and modifications may be effected therein by one skilled in that art without departing from the scope of the invention.